University of Saskatchewan
Department of Computer Science

CMPT 214 - Programming Principles and Practices

FINAL EXAM
December 15, 2004

160 Marks
Time: 180 Minutes
(OPEN BOOK. NO CALCULATORS)

Student's Name: _____
(Please print in BLOCK letters, surname first.)

Student Number: _____

Instructor's Name: David J. Callele and Dwight J. Makaroff

Special Instructions: The weight of each question is given in parentheses. The total number of marks is 160. The total time is 180 minutes. Budget your time.
This examination consists of 12 pages, including this cover page. **Check to ensure that this exam is complete.**

1. READ AND OBSERVE THE FOLLOWING RULES:
   - Answer each of the following questions in the space provided in this exam booklet. If you must continue an answer (e.g.in the extra space on the last page, or on the back side of a page), make sure you clearly indicate that you have done so and where to find the continuation. If you should find it necessary to do so, you have probably written too much.
   - Make all written answers legible; no marks can be given for answers which cannot be decrypted. Where a discourse or discussion is called for, be concise and precise.
   - If you find it necessary to make any assumptions to answer a question, state the assumption with your answer.
   - For explanation questions, please *USE YOUR OWN WORDS.* Answers taken directly from a textbook, or the notes will receive a grade of 0 (zero).
   - ALWAYS SHOW ALL YOUR WORK. A final answer, even if correct, without supporting work is not acceptable.

P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	12	TOTAL
10	25	18	12	20	10	22	10	18	14	4	160

1. (10 marks) Write a *bash* shell script called *validate* that takes, as a single command-line argument, a single file name. Each line of the input file is *expected* to have data in the format:

   ```
   name,grade
   ```

   Your shell script is to read the input file and ensure that each line in the input file is of the expected form. The *name* field can be composed of any characters that are not numbers in the range 0 to 9. The *grade* field can be composed of integers in the range 0 to 100 inclusive.

   The script *validate* must be constructed in a manner consistent with other utility scripts (and applications) that can be concatenated using the pipe (|) operator.

2. (20 marks) Given the following data set, give the regular expressions to match the defined patterns.

```
FirstModel.mdl
FirstModel_01.mdl
SecondModel_00.mdl
ThirdModel_01.mdl
FourthModel00.mdl
Fourthmodel_01.mdl
FourthModel_02.mdl
FifthModel_00.mdl
FifthModel_01.mdl
```

   - All strings that start with the letter F

   - All strings containing the sequence 01 or higher (e.g. 02, 03, etc.)

   - All strings that do not contain the string Model

   - All strings that do not contain a sequence number in the form _xx where x is a digit.

3. (5 marks) In the context of the bash shell, what is *command substitution*? Give an example that demonstrates how command substitution works and explain what your example does.

4. (5 marks) What is the difference between a *reference* in Java and a *pointer* in C?

5. (5 marks) Explain the relationship between *pointers* in C and the array construct [ ] in C. How are they the same? Different?

6. (8 marks) We had two guest speakers in class this term. The speakers identified many *non-technical* skills that were necessary for success in their environments. Give two examples of these skills and why they are important.

7. (5 marks) Why is it beneficial to be involved in a software consortium? Choose 2 reasons and explain in your own words.

8. (3 marks) When using a Makefile, how does *make* determine which actions to carry out?

9. (4 marks) How are *patterns* in *awk* and *targets* in a *Makefile* the same? How are they different?

10. Functions and function pointers.

    (a) (10 marks) Write a function *vowelUP*that maps all lower-case vowels (a,e,i,o,u) in a given string to their uppercase equivalents (A, E, I, O, U). The function prototype is:

    ```
    char *vowelUP(char *cptrTheString);
    ```

    (b) (10 marks) Assume that you also have a function *vowelDOWN*that maps all upper-case vowels (A, E, I, O, U) in a given string to their lower-case equivalents (a,e,i,o,u). The function prototype is:

    ```
    char *vowelDOWN(char *cptrTheString);
    ```

    Also assume that you have a function pointer MyFunctionPtr that can be used to point to functions of this type (*vowelUP and vowelDOWN*, a flag variable called *myFlag* and a string called *myString*. Give a code fragment that:

    i. initializes MyFunctionPtr to point to *vowelUP*
    ii. if myFlag is TRUE sets MyFunctionPtr to point to *vowelDOWN*
    iii. using *MyFunctionPointer*, modifies *myString*

11. A phone book stores records of the following type:

```
Callele DJ, 25 Preston Ave, Saskatoon, SK, 3065551212
Makaroff DJ, 1 Main St, Calgary, AB, 4034567890
```

a) (5 marks) Define a User Defined Type called PHONE, based on a C struct, that could be used to store the information for a single record. You may assume a reasonable fixed width for each field.

b) (5 marks) If you are *not* allowed to assume a reasonable fixed width for each field, explain how you would modify PHONE so that you could handle variable length fields. Are further data structures necessary?

12. (6 marks) Write an *awk program* to determine and print out how many times column 2 is numerically greater than column 3 in a text file with the following format (comma is the delimeter):

```
''Mario Lemieux'',54,36,90
''Joe Sakic'',27,49,76
''Markus Naslund'',51,48,99
```

13. (4 marks) Explain (in your own words) the difference between

    #define DEBUG 1 and

    const int DEBUG = 1 and how this is handled by the compiler.

14. (6 marks) Explain (in your own words) how a user regains keyboard control of a process that has been put into the "background"? Explain 2 (two) examples of how a process can be put into the "background".

15. (6 marks) Explain (in your own words) why Perl and the Web are good for each other.

16. (3 marks) How is a hash table different from an array in Perl.

17. (7 marks) Write a Perl program to process a text file as follows:

    Calculate the number of times a particular *word* occurs in the text file. The word to look for is provided as the first command-line argument. A word is considered any non-space characters separated by whitespace characters (or the newline character).

18. (18 marks) The following code sample has (at least) 3 places where a run-time failure could occur. Identify **three (3)** errors (circle them in the source code). Suggest alternative code and justify your suggestion. Finally, suggest a test case for each error that would identify the original error.

```
int scale(int* ipInput, int* iCols, int iScaleFactor)
{
  int x = 0;
  for (x = 0; x <= iCols; ++x)
  {
     ipInput[x] = *(ipInput + x) / iScaleFactor;
  }
  return(&ipInput);
}
```

19. (6 marks) When building a library, identify 3 considerations the implementor must make with respect to the user and why these considerations are important.

20. (8 marks) Explain briefly, using your own words:

    - How processes are created in an operating system? (You may make reference to the UNIX operating system, but a correct answer does not depend on UNIX details)

    - Describe 2(two) components of the memory image of a process and how the*use* of these components differs from each other.

    - Explain how processes are terminated.

21. (4 marks) Give 1 (one) reason to prefer dynamic linking over static linking and 1 (one) reason to prefer static linking over dynamic linking.

THE END

The remainder of this sheet is for rough work.